Ruby trunk - Feature #15765

[PATCH] Module#name without global constant search

04/12/2019 04:12 AM - alanwu (Alan Wu)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Hello!

The current implementation of Module#name is known for having sub-optimal performance when it comes to anonymous modules. (see #11119 and #15625)
I have put together a reimplementation of Module#name, which works by eagerly naming
modules and classes when they are assigned to constants. Besides solving the performance
issues for Module#name on anonymous modules, there are some other benefits.

This patch:

- removes more code than it adds
- makes normal class and module definition slightly faster (definitions like class Foo; end)
- slightly reduces memory usage for classes and modules due to the removal of a hidden ivar
- improves the performance of defining modules and classes under an anonymous module. This used to execute a global search each time.

Behavior changes and caveats:

Since we already name module and classes declared with the class and module keyword on trunk, this patch mostly targets anonymous
modules. I tried my best keeping the behaviors consistent with the current implementation, but there are some small behavioral changes.

```
mod = Module.new
mod::BeforeToS = Module.new
mod.const_set(:BeforeToS2, Module.new)

mod.to_s # on trunk, the VM starts naming modules assigned under mod after calling to_s

mod::AfterToS = Module.new
mod.const_set(:AfterToS2, Module.new)

p mod::BeforeToS.name  # nil on both
p mod::BeforeToS2.name # nil on both
p mod::AfterToS.name   # "#<Module:0x0000563494b1cca8>::AfterToS" on trunk, nil after patch
p mod::AfterToS2.name  # "#<Module:0x0000563494b1cca8>::AfterToS2" on trunk, nil after patch
```

This prints 4 nils after my patch, as I think the behavior on trunk is unintentional. A few C APIs also have the same effect as calling to_s. They are all changed to be side-effect free.

```
m = Module.new
m::Child = Module.new
Mod = m
p Object.send(:remove_const, :Mod)::Child.name
```

This prints nil on trunk and Mod::Child under this patch.

rb_name_class is removed, as it does nothing in this new implementation. Not sure if this is public API.

Since the recursive naming is done with a recursive function, when a deeply nested anonymous module is assigned
to a constant, it is technically possible for this implementation to throw a StackError. I had a version
which does heap allocation to deal with this, but I picked this version for performance in the common cases.
Anonymous modules are rare as is, and one would have to build a structure nested thousands level deep for this to happen.

On my system it can name a module fifty thousand levels deep without problem.

I think these changes are fairly minimal and acceptable.

History

#1 - 04/12/2019 04:17 AM - alanwu (Alan Wu)

This is for #11119 and #15625.

#2 - 04/17/2019 12:42 AM - alanwu (Alan Wu)

- Description updated

Files

benchmarks.rb	3.16 KB	04/12/2019	alanwu (Alan Wu)
0001-Eagerly-name-modules-and-classes.patch	19.8 KB	04/12/2019	alanwu (Alan Wu)